STAT572 Midterm 1
Jon Braswell
4/10/2020

Problem 1a:

Use the alternative CDF method to generate a random sample of size 200 from the density. Plot the density histogram of the random sample. Also provide the empirical and theoretical CDF and compare.

Procedure

The first step is to define a range of x-values to plug into my distribution. Looking at the Slash distribution I chose to use X = [-20,20]. Next I wrote a function that would return the CDF of the theoretical distribution. To do this I have the function return the integral of the given function from negative infinity to the input argument, unless the input argument was zero, in which case the function would return 0.5, as that is the center of the distribution. I then applied this function to each of my X values to create another vector that would represent each point's respective cumulative probability. Finally a loop creates a random number from [0,1] and then finds the X whose CDF value is closest to that random uniform number, this X is added to the sample. This loop is repeated until I have the required sample.

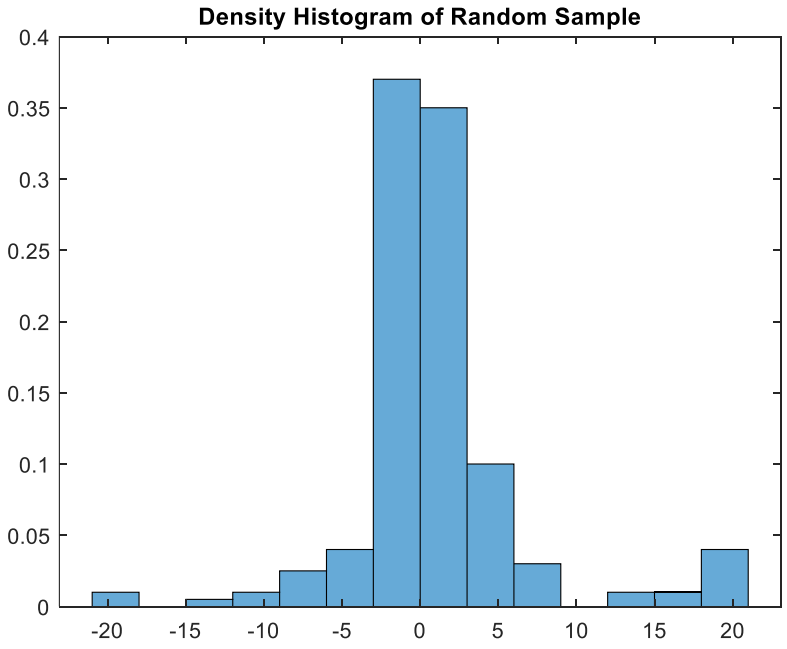
Matlab Code:

```
function [out] = slash_cdf(in)
if in == 0
    out = .5;
else
    out = integral(@(x)((1-exp(-x.^2/2))/(x.^2*sqrt(2*pi))),...
        -inf,in, 'ArrayValued',true);
end


% Part One

clear all;close all;
n = 200;
step=.1;, range=[-20,20];
i=(range(2)-range(1))/step;
xs=linspace(range(1),range(2),i);
ys=arrayfun(@(a) ((1-exp(-a.^2/2))/(a.^2*sqrt(2*pi))),xs);

samp = zeros(1,n);
cum_dens = arrayfun(@(a) slash_cdf(a),xs);
```

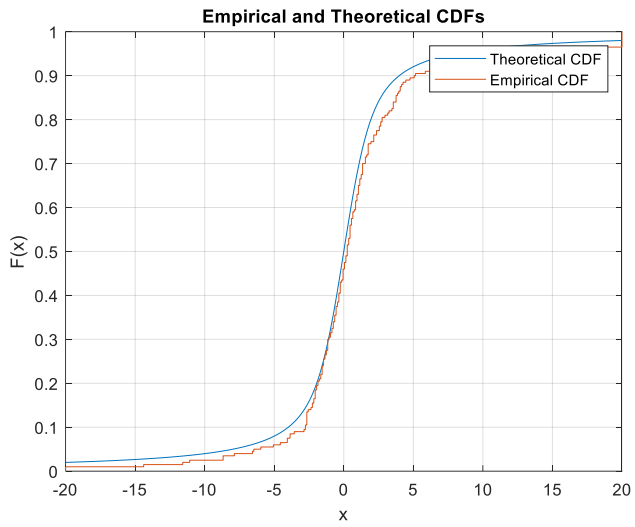
```
for i = 1:n
    u = rand(1);
    [k,dist] = dsearchn(cum_dens',u);
    samp(i) = xs(k);
end

figure
histogram(samp,'Normalization','probability')
title('Density Histogram of Random Sample')
figure
plot(xs,cum_dens)
hold on
cdfplot(samp)
hold off
legend('Theoretical CDF', 'Empirical CDF')
title('Empirical and Theoretical CDFs')
```

Output:

Plot the density historgram of the random sample.



Also provide the empirical and theoretical CDF and compare.



Discussion:

The density histogram, with its tall narrow center and low, fat tails is a very close replica of the theoretical Slash distribution. When we look at the theoretical and empirical CDFs we also see a very similar shape.

Problem 1b:

Using the random sample in a), we are interested in a Monte Carlo test for

H0: m = 1 and H1: m!=1

Where m is the population median. Carefully describe how you would perform the Monte Carlo test. Calculate the Monte Carlo *p-value* and draw a conclusion

Procedure

1. Calculate the value of the test statistic using a random sample (from above) of size n from the population. This test statistic is the median.
2. Determine a pseudo-population. A Q-Q plot is created with the data and it is concluded that in the center the data is close enough to a N(1,1) to use that as a distribution to come up with the pseudo-population.
3. Obtain a random sample from your pseudo-population under H0. H0 is that the median is equal to 1. If the pseudo population is Normal, then a Normal distribution with a median of 1 would be N(1,1). This is the distribution from which I will sample.
4. Draw a sample and calculate the test statistics (median)
5. Repeat 3 and 4 M many times. In my case I chose 100 trials.
6. Using the M values from the pseudo-population, determine the rejection region. This is a two tailed test at 95% confidence, so we choose all those medians that are above .025 percentile and below the .975 percentile. The values in this range form a 95% confidence region.
7. The median of our sample fell outside of the confidence interval, so the null hypothesis was rejected.

Matlab Code:

```
% Part Two
med = median(samp);
alpha = .05
% 2. Determin a Psuedo Population
normplot(samp)
% Data is normal around the center and so should share a median with a
% normal distribution.  We will use Normal distribution to generate psuedo
% population.

mc = 1000
medians = zeros(1,length(mc))
for i = 1:mc
% 3. Obtain a random sample of size N from the pseudo-population under H0.
% H0: median = 1.  To generate this sample we will use N(1,1).
    mc_data = normrnd(1,1,1,length(samp));
% 4.  Calculate the median for each of these samples.
```

```
    medians(i) = median(mc_data);
% 5. Repeat 3 and 4 M many times.
end

% 6. Using the M values from the pseudo-population, determin the rejection
% region.
ub = csquantiles(medians,alpha/2)
lb = csquantiles(medians,1-alpha/2)
fprintf('The Upper Bound is %f\n', ub);
fprintf('The Lower Bound is %f\n', lb);

if ((med >= lb) && (med<=ub))
    fprintf('Accept the Null Hypothesis')
else
    fprintf('Reject the Null Hypothesis')
end
```

Output:

```
The Upper Bound is 0.825677
The Lower Bound is 1.176618
Reject the Null Hypothesis>>
```

Discussion:

From our pseudo-population with a median of 1 we determined that a 95% confidence bounds for the median would be between 0.82 and 1.17. The median from the sample was 0.25, which fell outside of the confidence bounds. Thus we reject the null hypothesis that the underlying population as a median of 1.

Problem 2:

Use an acception-rejection method to generate a random sample of size 300 from the density. For candidate distributions use Unif(2,6). Provide the density histogram of the random samples with the true density superimposed. Give the figure showing the accepted and rejected variates as in Figure 4.3 and calculate the probability of rejection.

Procedure

The first step is to choose a candidate distribution from which to sample. This is chosen to be the Uniform[2,6], as it's domain is the same as the target distribution. The maximum value of the target distribution is .5, and the maximum of the candidate distribution is .25 - so a value of c = 2 is chosen to ensure that the target distribution is below the candidate distribution.

For each iteration of the algorithm two standard uniform random variables are chosen. These are used to create the cut off and the value of the function. A step wise function is used to separate x values of less than 3, and more than 3 – applying a different function depending.

If the generated value is less than the function's value it is accepted and we keep it as well as generated density. We keep doing this until we have 300 accepted random variables.

After plotting these we calculate the probability of rejecting by finding out of the sample points that were generated, how many of them were rejected.

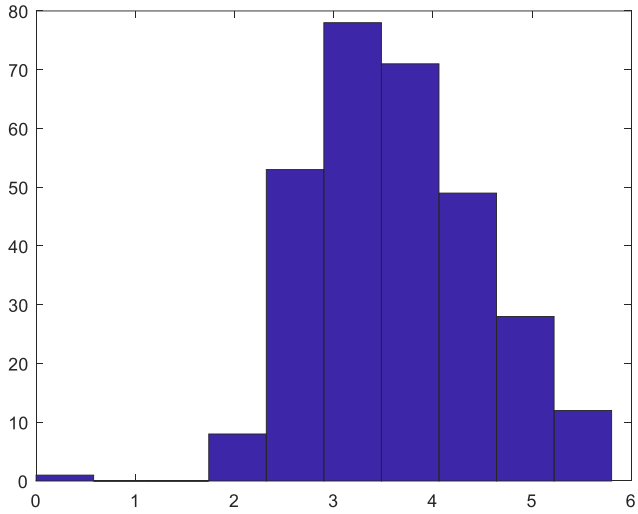
Matlab Code:

```
clc;
clear all;close all;
n=300; a=2; b=6;
ymax=0.5; n = 300;
x = zeros(1,n);xy = zeros(1,n);
rej = zeros(1,n);rejy = zeros(1,n);
irv=1; irej=1;
q=0;
while irv < n
    u1=rand(1,1);
    u2=rand(1,1);
    x1=a+u1*(b-a);
    y1=ymax*u2;

    if x1 < 3
        f=(x1-2)/2;
    else
        f=(2-(x1/3))/2;
    end
```

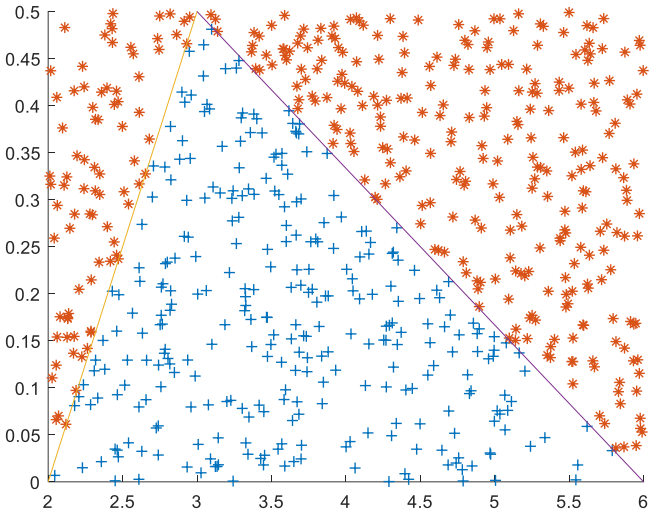
```
    if y1<=f
         x(irv)=x1;
         xy(irv)=y1;
         irv = irv + 1;
         q = q + 1;
    else
        rej(irej) = x1;
        rejy(irej) = y1;
        irej = irej +1;
        q = q + 1;
    end
end
hold on
plot(x,xy,'+')
plot(rej,rejy,'*')
fplot(@(x) (x-2)/2, [2 3])
fplot(@(x) (2-(x/3))/2, [3 6])
xlim([2 6])
hold off
```

Output:

Histogram of Sample



Plot of Accepted/Rejected Points



Probability of Rejecting

```
rej_prob =

    0.5427
```

Discussion:

The histogram of the data shows a higher density of points at 3 and above. Looking at the original density function this makes sense, as 75% of the density is to the right of 3.

Looking at our plot of accepted and rejected points we see that it makes the exact triangle shape, with its apex at (3,0.5). Whenever a point's density was below that of the target function, we kept that point and added it to our sample, this created a sample that approximated the given distribution.

The probability of rejecting a point with this method was .54, a probability that is low enough to make this algorithm perform in little time.

Problem 3:

Suppose we want to estimate θ, where

$$\theta = \int_0^1 e^{x^2} dx$$

We want to argue that generating a random variable U from $Uniform(0,1)$ and then using the estimate $e^{U^2}(1+e^{1-2U})/2$ is better than generating two uniform random numbers and using $(e^{U_1^2}+e^{U_2^2})/2$. Demonstrate this by a simulation.

Procedure

Generate three random numbers from U(0,1). Use these numbers to generate the two candidate estimates. Perform this 1000 times and store all the results. Use these results, and the actual value of the parameter to estimate the MSE.

Matlab Code:

[Your code here]

Output:

```
mc = 1000;
est1 = zeros(length(mc));
est2 = zeros(length(mc));
true_val = integral(@(a) exp(a.^2),0,1);
for i = 1:mc
    u = rand(1);
    w = rand(1);
    v = rand(1);
    est1(i) = (exp(u^2)*(1+exp(1-2*u)))/2;
    est2(i) = ((exp(w^2)+exp(v^2)))/2;

end
mse_est1 = mean((est1-true_val).^2);
mse_est2 = mean((est2-true_val).^2);
```

```
mse_est1 =

    0.0292

>> mse_est2

mse_est2 =

    0.1120
```

Discussion:

The MSE for the first estimate, that uses only one uniform random variable, is significantly lower than that of the second estimate. Therefore the first estimate is considered better.